Python Data Types

- Python

Data Science: Python Data Types

Integer numbers	int	100
Float, real numbers	float	100.0
Text, string	str	“data science” ‘data science’
Boolean	bool	True or False

Data Science: Python Data Types

Array		
Dictionary (unordered, changeable, indexed, no duplicates) (Can access items by ["key"])	dict	{"course": "Data Science Fundamentals", "mode": "Online", "term": "Fall 2019" }
Lists (ordered, changeable, duplicates are ok) (can access items by [index#])	list	["Adam", "Betsy", "Cherry", "Betsy"]
Tuples (ordered, unchangeable, duplicates ok) (can access items by [index#])	tuple	("may", "june", "july","july", August)
Sets (unordered, unindexed, no duplicates) (cannot access items by index#)	set	{"apple", "banana", "craneberry"}

Dynamic Arrays in Data Structures

- In the class for a data structure, we can add an Array object to the private section to store data
- The functions of the data structure can expand or shrink the Array to conserve memory – this relieves the client from thinking about doing this

Arrays in
Data Structures

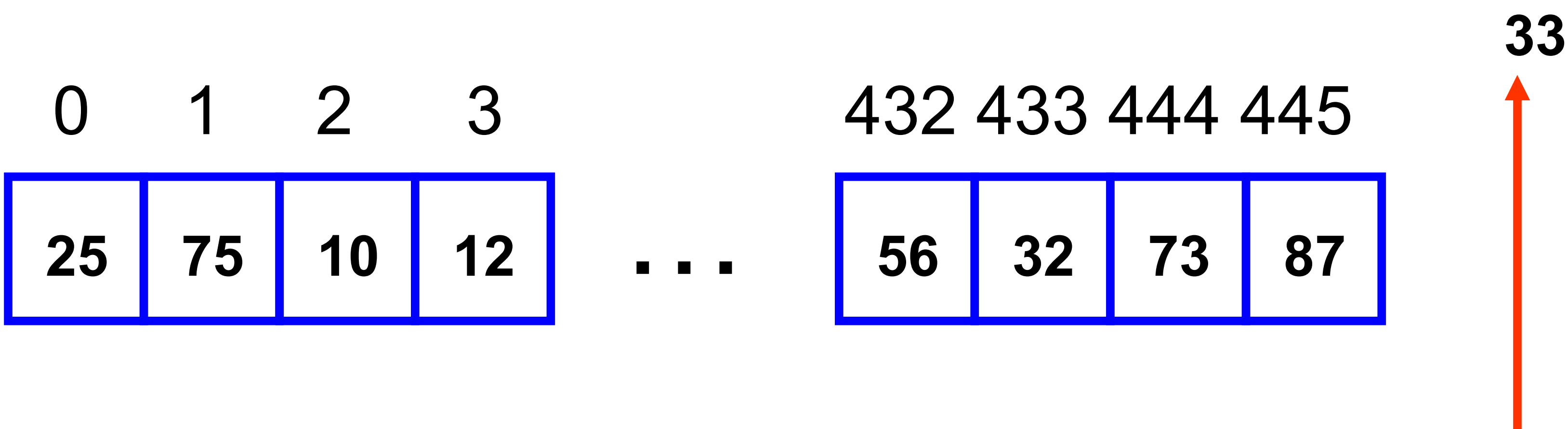
Dynamic Arrays in Data Structures (cont.)

- In almost every data structure, we want functions for inserting and removing data
- When dynamic arrays are used, the insertion function would add data to the array, while the removal function would "eliminate" data from the array (make it unusable)
- When the array becomes full, we would want to do an expansion – when many elements have been removed, we would want to do a contraction, so that only the used elements remain

Array Expansion/Contraction

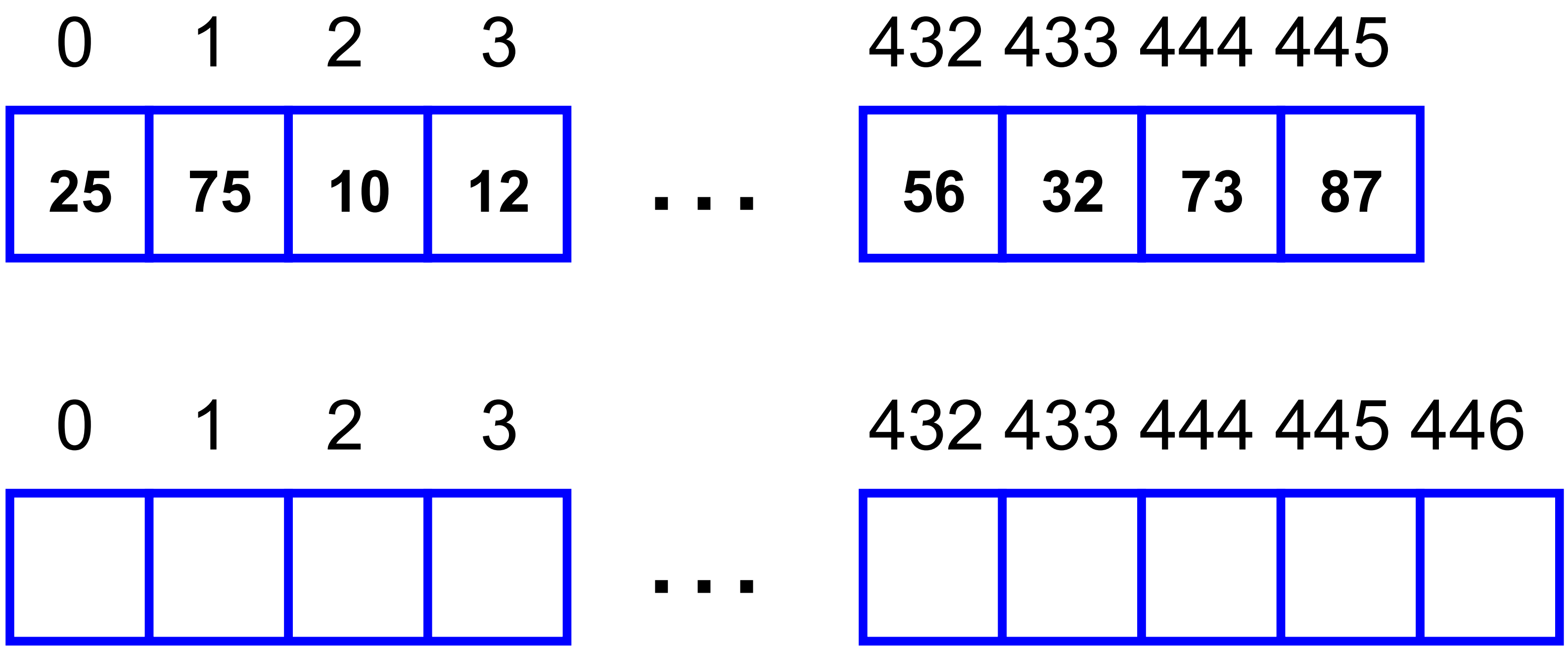
- One possible method:
  - When an element is inserted by the client, increase the size of the array by 1
  - When an element is removed by the client, decrease the size of the array by 1
- The problem with this method is that it is inefficient – every time an element is inserted or removed, the changeSize function is called…

changeSize Function



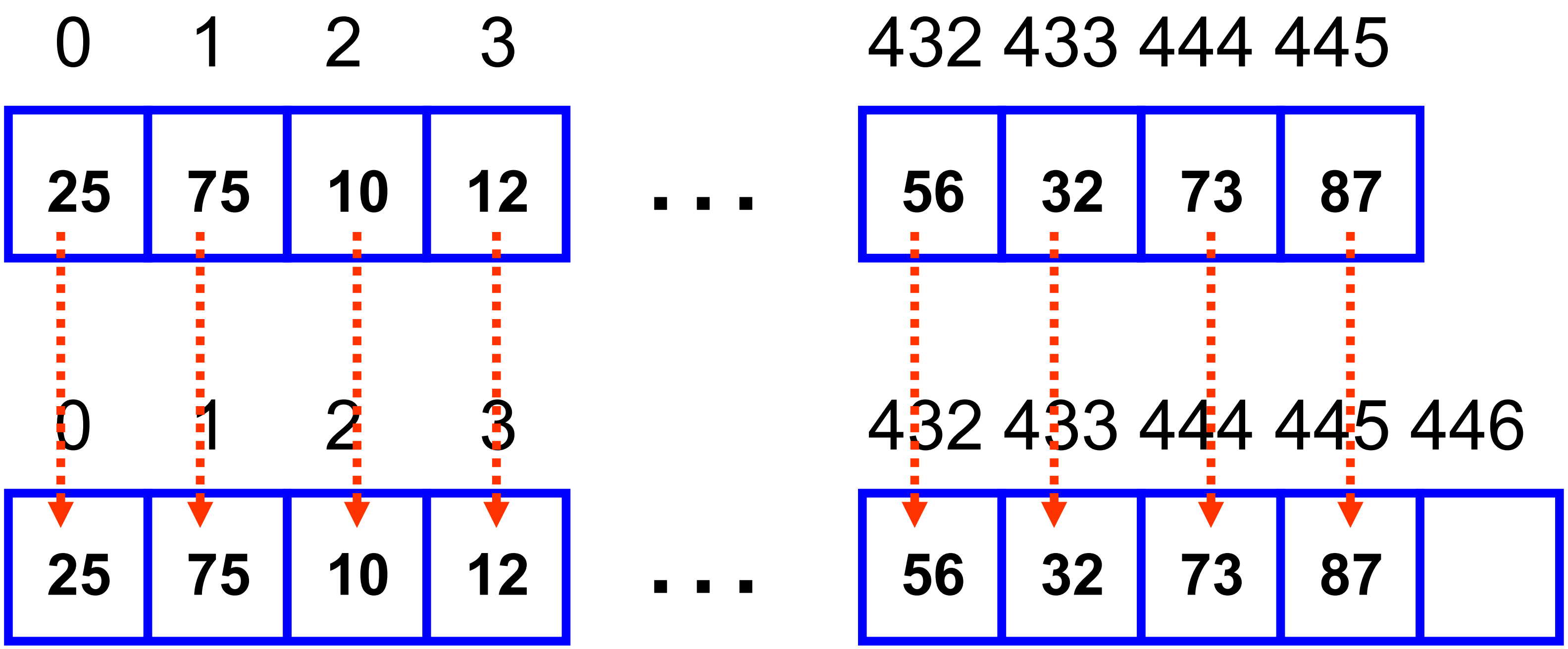
New element needs to be put into array, so changeSize function is called

changeSize Function (cont.)



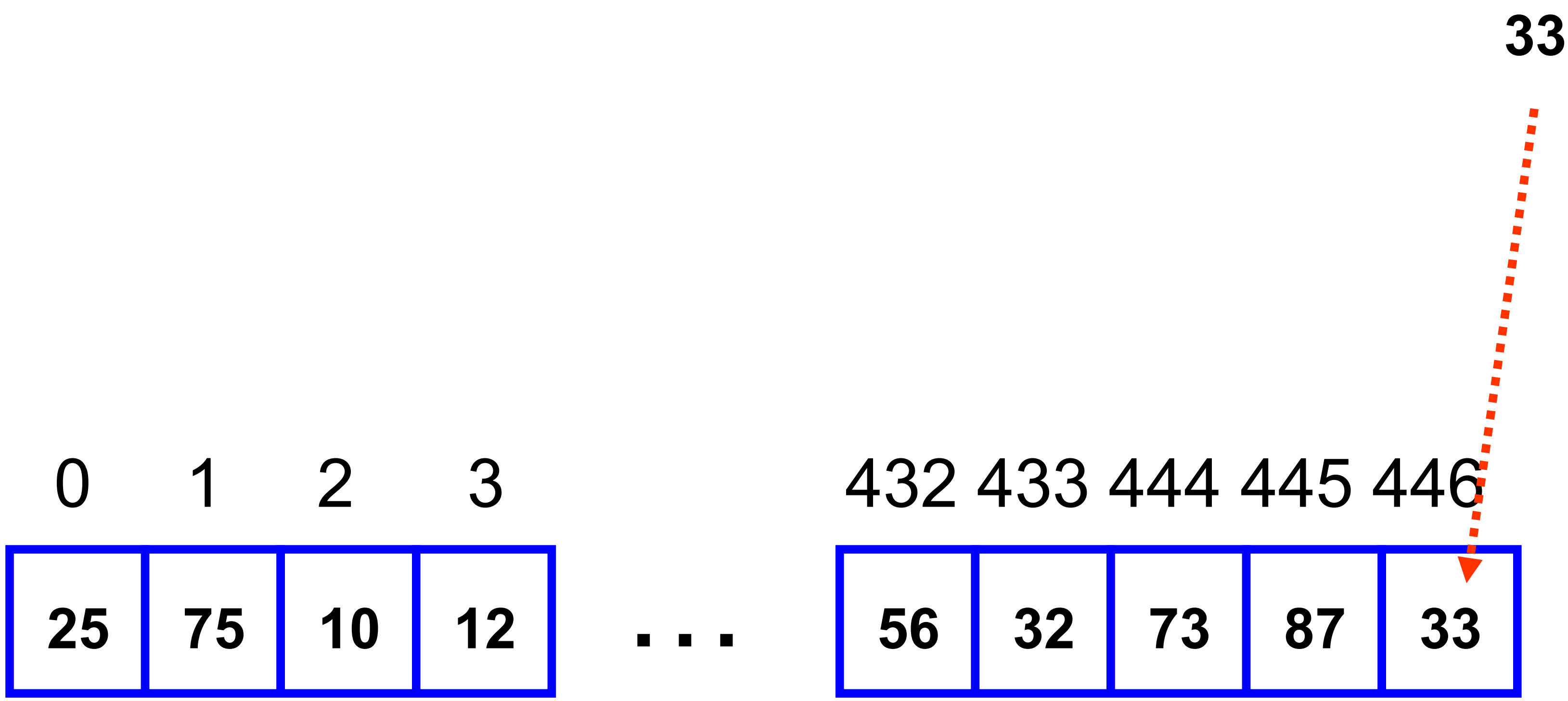
new array is made

changeSize Function (cont.)



elements are copied over one by one using a for loop

changeSize Function (cont.)



Then, the new element can be put in

changeSize Function (cont.)

0	1	2	3		432	433	444	445	446
25	**75**	**10**	**12**	**. . .**	**56**	**32**	**73**	**87**	**33**

This process would take place every time a new element needs to be inserted.

changeSize Function (cont.)

0	1	2	3		432	433	444	445	446
25	**75**	**10**	**12**	**. . .**	**56**	**32**	**73**	**87**	**33**

Likewise, when an element needs to be removed, this method contracts the array by one to conserve memory.

changeSize Function (cont.)

0	1	2	3		432	433	444	445	446
25	**75**	**10**	**12**	**. . .**	**56**	**32**	**73**	**87**	**33**

Suppose the element at the end of the array needs to be removed.

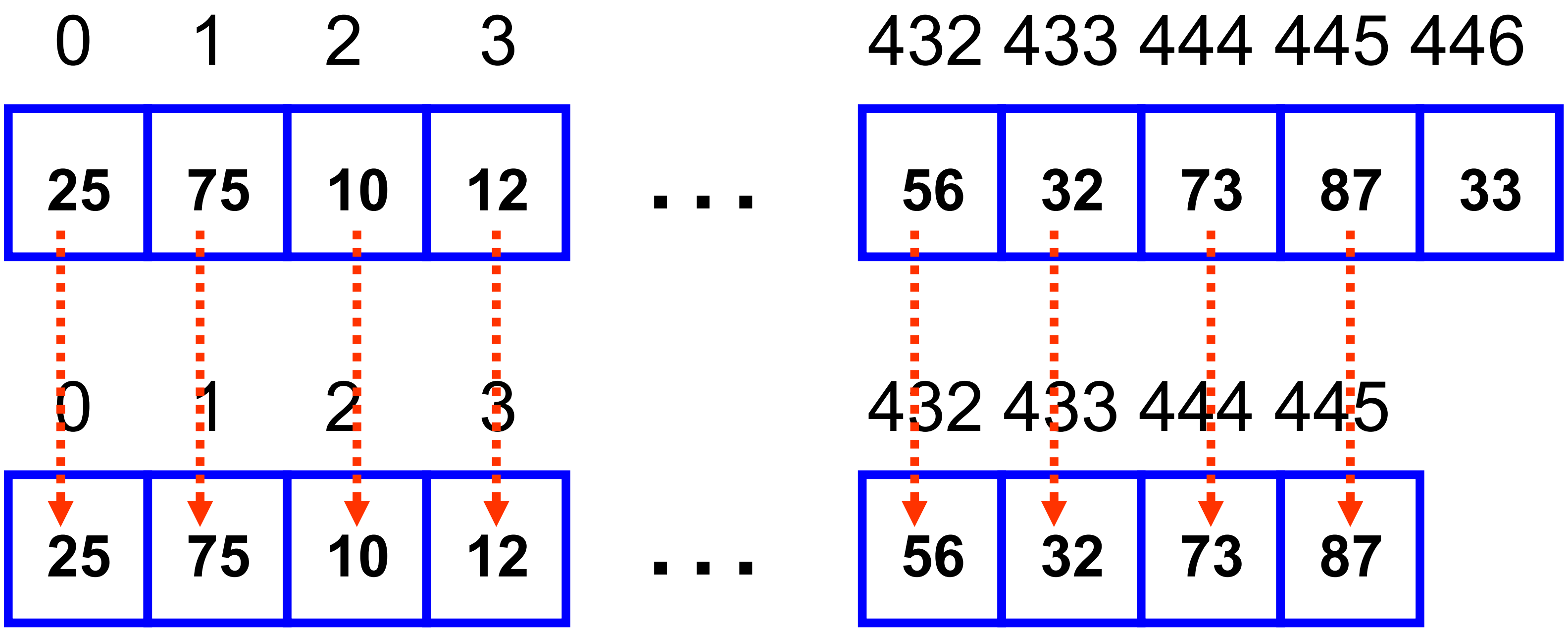
changeSize Function (cont.)

0	1	2	3		432	433	444	445	446
25	**75**	**10**	**12**	. . .	**56**	**32**	**73**	**87**	**33**

0	1	2	3		432	433	444	445
				. . .				

The changeSize function is called and a new, smaller array is made.

changeSize Function (cont.)



The elements are copied over one by one, using a for loop.

changeSize Function (cont.)

0	1	2	3		432	433	444	445
25	**75**	**10**	**12**	**. . .**	**56**	**32**	**73**	**87**

This method of array expansion/contraction is largely inefficient, because there is too much element copying.

A Better Method

- When the Array is full, double the size of it
- When the number of elements used in the Array falls to 25% of the Array’s capacity, cut the size of the Array in half (it will be half full after the cut)

A Better Method (cont.)

0	1	2	3
25	**75**		

A Better Method (cont.)

0	1	2	3
25	**75**	**10**	

A Better Method (cont.)

0	1	2	3
25	**75**	**10**	**12**

A Better Method (cont.)

0	1	2	3
25	**75**	**10**	**12**

33

Array is full, so call changeSize function to double the size.

A Better Method (cont.)

0	1	2	3	4	5	6	7
25	**75**	**10**	**12**	**33**			

A Better Method (cont.)

0	1	2	3	4	5	6	7
25	**75**	**10**	**12**	**33**	**49**	**29**	**87**

This array is full, but if we removed elements (made them inaccessible), we would cut the size of the array in half when its utilization drops to 25%

A Better Method (cont.)

0	1	2	3	4	5	6	7
25	**75**	**10**	**12**	**33**	**49**	**29**	

A Better Method (cont.)

0	1	2	3	4	5	6	7
25	**75**	**10**	**12**	**33**	**49**		

A Better Method (cont.)

0	1	2	3	4	5	6	7
25	**75**	**10**	**12**	**33**			

A Better Method (cont.)

0	1	2	3	4	5	6	7
25	**75**	**10**	**12**				

A Better Method (cont.)

0	1	2	3	4	5	6	7
25	**75**	**10**					

A Better Method (cont.)

0	1	2	3	4	5	6	7
25	**75**						

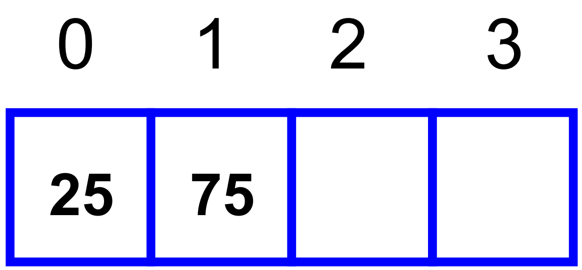
Array is 25% utilized, so use changeSize function to cut the size of the array in half.

A Better Method (cont.)

0	1	2	3
25	75		

Array is 25% utilized, so use changeSize function to cut the size of the array in half.

A Better Method (cont.)



Using this method, memory is still conserved.

There is element copying every time changeSize is called, but it isn't bad.

A Better Method (cont.)

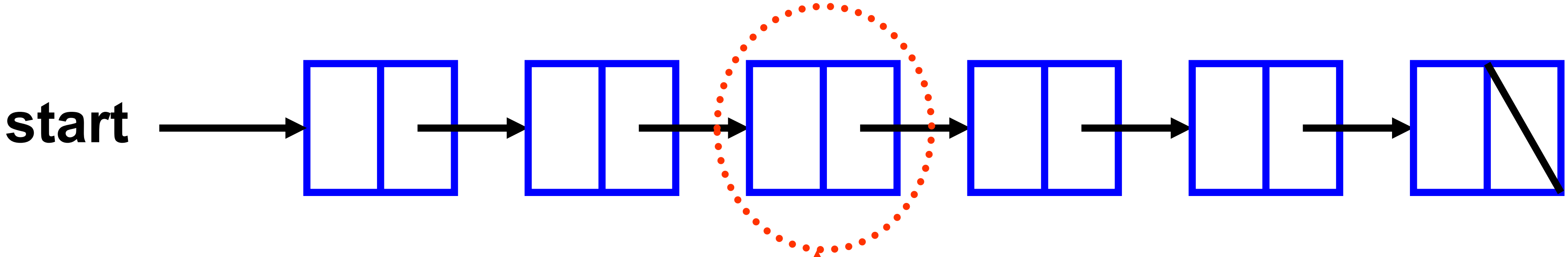
0	1	2	3
25	**75**		

It can be proven that, on average, there are no more than a couple of elements being copied on each insertion/deletion with this method.

Linked Structures

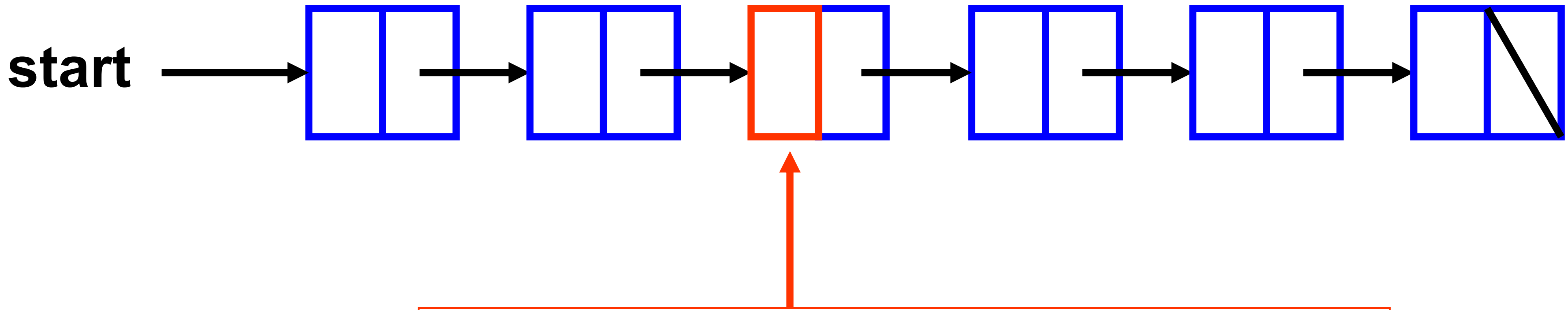
- In a data structure, data is not always stored in an Array object
- Sometimes it is best to store data in a ***linked structure*** (an alternative to an Array)
- A linked structure consists of a group of ***nodes*** – each node is made from a struct.
- An object of the Node struct contains an element of data.

Example of a Linked Structure



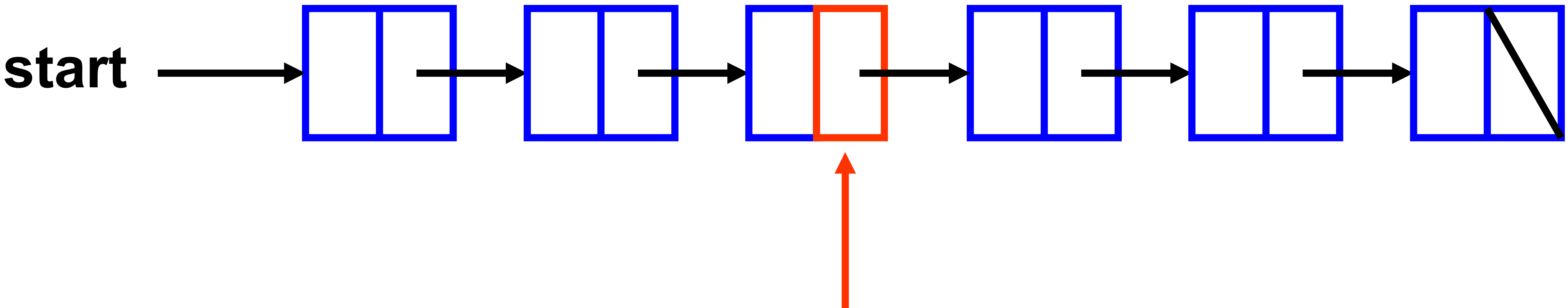
Each blue node is divided into two sections, for the two members of the Node struct.

Example of a Linked Structure (cont.)



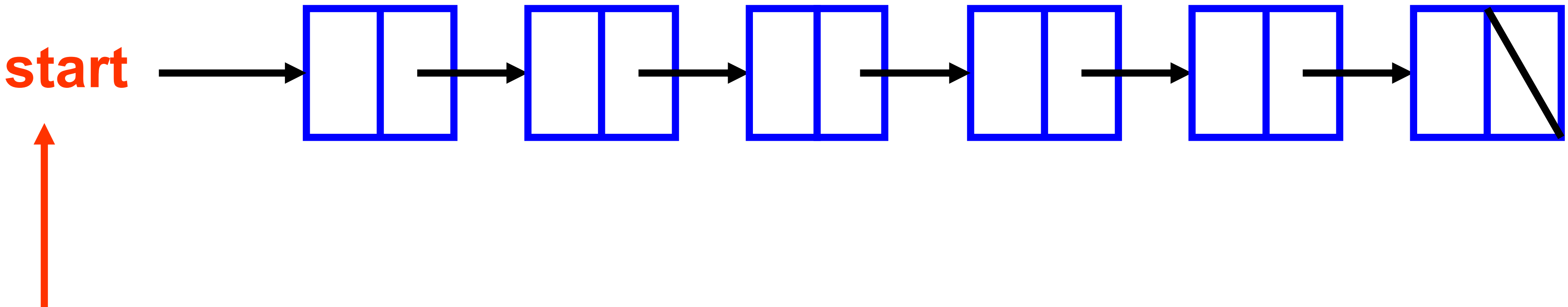
The left section is the data member.

Example of a Linked Structure (cont.)



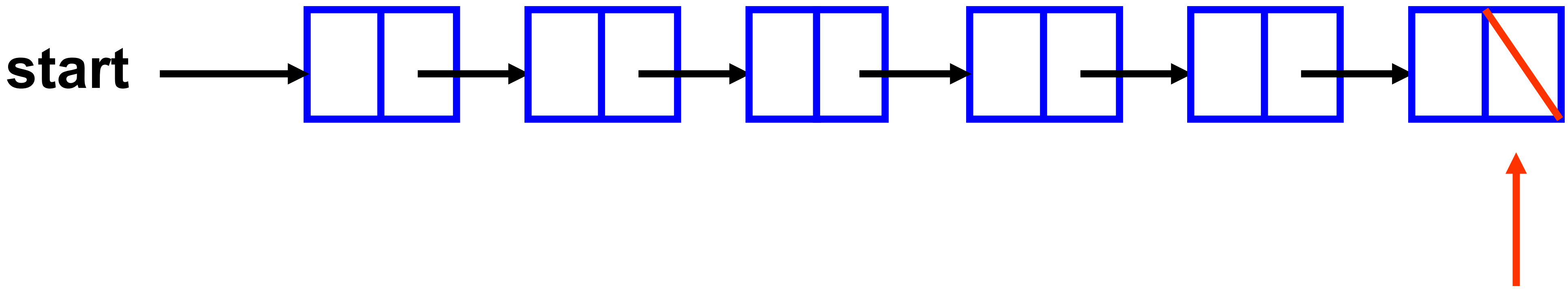
The right section is the pointer called “next”.

Example of a Linked Structure (cont.)



The start pointer would be saved in the private section of a data structure class.

Example of a Linked Structure (cont.)

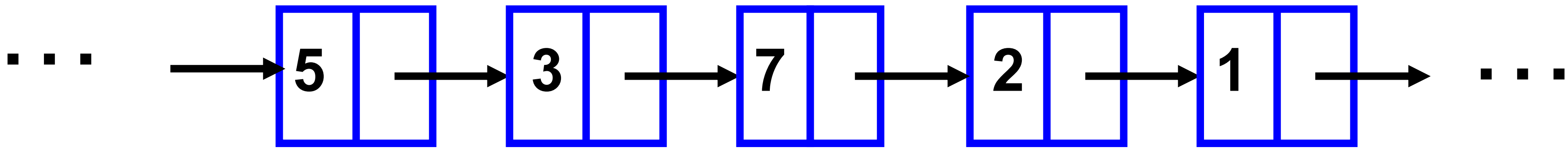


The last node doesn't point to another node, so its pointer (called next) is set to NULL (indicated by slash).

Linked Lists

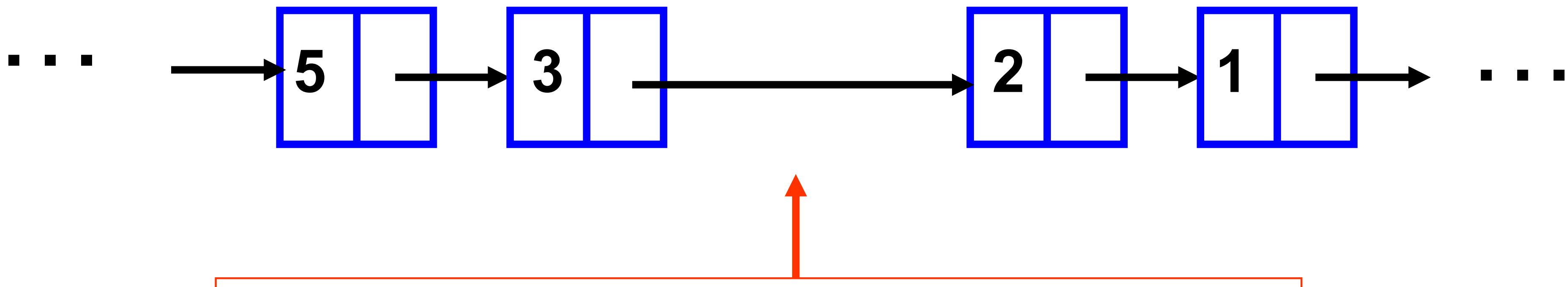
- The arrangement of nodes in the linked structure on the previous slide is often called a ***linked list***.
- We can access any element of the linked list, for retrieval of information.
- We can also remove any element from the linked list (which would shorten the list).
- We can also insert any element into any position in the linked list.

Linked List Advantages



Removing an element from the middle of a linked list is fast.

Linked List Advantages (cont.)



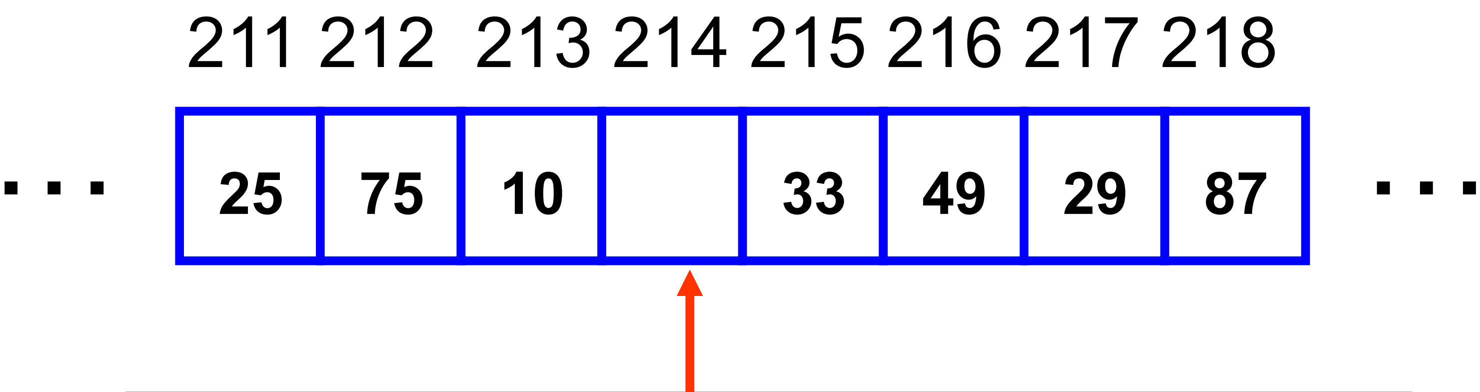
Removing an element from the middle of a linked list is fast.

Linked List Advantages (cont.)

	211	212	213	214	215	216	217	218	
. . .	**25**	**75**	**10**	**12**	**33**	**49**	**29**	**87**	. . .

Removing elements from the middle of an array (without leaving gaps) is more problematic.

Linked List Advantages (cont.)



Removing elements from the middle of an array (without leaving gaps) is more problematic.

Linked List Advantages (cont.)

	211	212	213	214	215	216	217	218	
. . .	**25**	**75**	**10**		**33**	**49**	**29**	**87**	. . .

A loop must be used to slide each element on the right one slot to the left, one at a time…

Linked List Advantages (cont.)

	211	212	213	214	215	216	217	218	
. . .	**25**	**75**	**10**	**33**		**49**	**29**	**87**	. . .

Linked List Advantages (cont.)

	211	212	213	214	215	216	217	218	
. . .	**25**	**75**	**10**	**33**	**49**		**29**	**87**	. . .

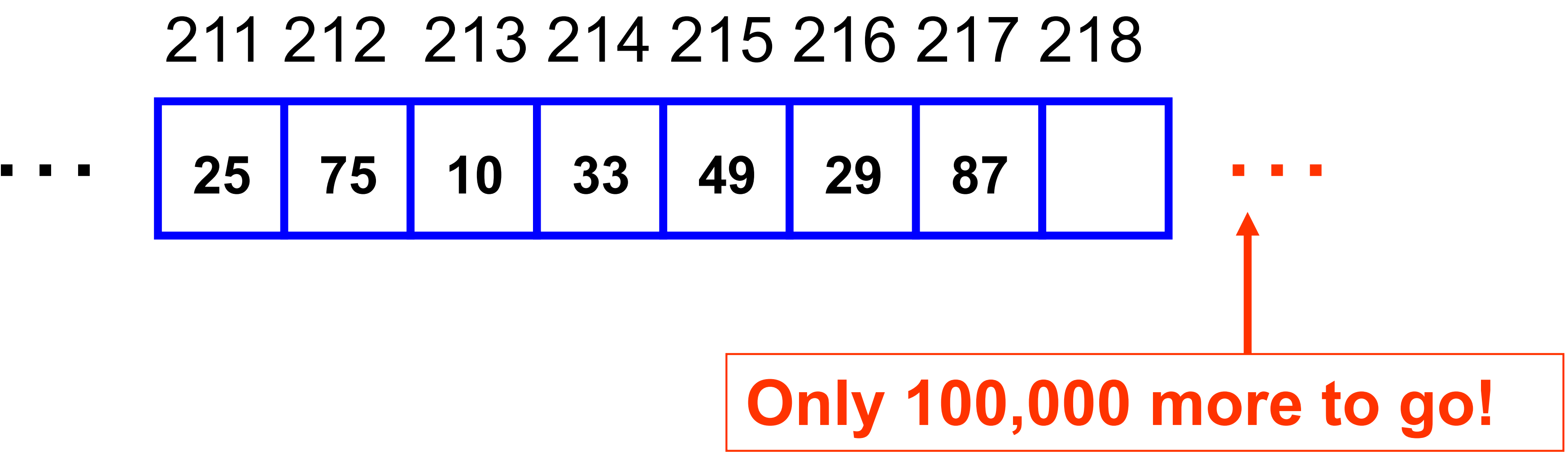
Linked List Advantages (cont.)

	211	212	213	214	215	216	217	218	
. . .	**25**	**75**	**10**	**33**	**49**	**29**		**87**	. . .

Linked List Advantages (cont.)

	211	212	213	214	215	216	217	218	
. . .	**25**	**75**	**10**	**33**	**49**	**29**	**87**		. . .

Linked List Advantages (cont.)



Python Data Types

- Numpy

Numpy Basics

- N-Dimensiomnal array data structure

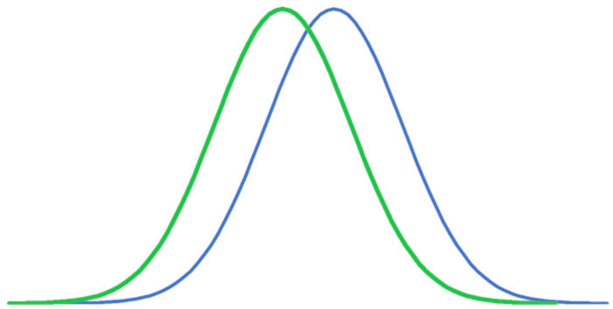
- comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, etc.

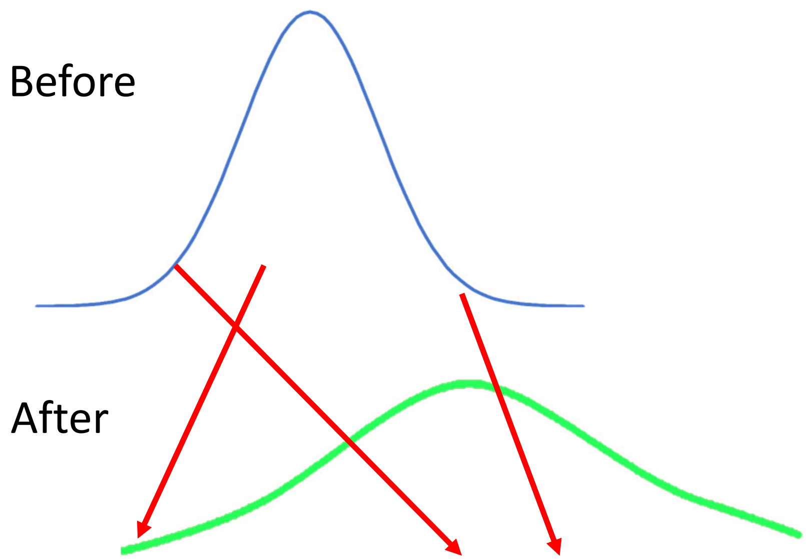
- Scientific computing

```
import numpy as np

# Create a 1-Dimensional array

temperature = np.array([60,70,80,90,100])
customer = np.array([100, 150, 180, 190, 195])

type(customer)

temperature.mean()

print("temperature data collected = ",temperature)
print("customer data collected = ", customer)

temperature[0]

# Correlation Analysis - Generate correlation matrix

np.corrcoef(temperature, customer)

Temp_Cust_2D = np.array([(60,70,80,90,100),(100, 150, 180, 190, 195)])

print (Temp_Cust_2D)

Temp_Cust_2D[0,2]

# Element wise operation - not possible on List data type

Temp_Cust_2D = Temp_Cust_2D*4

print (Temp_Cust_2D)

# Using Numpy Random number generator to generate a 2 dimensional array

TempCust = np.array([np.random.randint(50,100,20), np.random.randint(10,200,20)])
TempCust

correlMatrix=np.corrcoef(TempCust[0], TempCust[1])

print (correlMatrix)
```

T-Test

- A test of statistical differences




Before
After

Independent Sample T-Test

$$t = \frac{m_A - m_B}{\sqrt{\frac{S^2}{n_A} + \frac{S^2}{n_B}}}$$

m = mean
n = size
S^2 = estimator of the common variance of the two samples

$$S^2 = \frac{\sum (x - m_A)^2 + \sum (x - m_B)^2}{n_A + n_B - 2}$$

$$df = n_A + n_B - 2$$

Degrees of freedom	Significance level					
	20% (0.20)	10% (0.10)	5% (0.05)	2% (0.02)	1% (0.01)	0.1% (0.001)
1	3.078	6.314	12.706	31.821	63.657	636.619
2	1.886	2.920	4.303	6.965	9.925	31.598
3	1.638	2.353	3.182	4.541	5.841	12.941
4	1.533	2.132	2.776	3.747	4.604	8.610
5	1.476	2.015	2.571	3.365	4.032	6.859
6	1.440	1.943	2.447	3.143	3.707	5.959
7	1.415	1.895	2.365	2.998	3.499	5.405
8	1.397	1.860	2.306	2.896	3.355	5.041
9	1.383	1.833	2.262	2.821	3.250	4.781
10	1.372	1.812	2.228	2.764	3.169	4.587
11	1.363	1.796	2.201	2.718	3.106	4.437
12	1.356	1.782	2.179	2.681	3.055	4.318
13	1.350	1.771	2.160	2.650	3.012	4.221
14	1.345	1.761	2.145	2.624	2.977	4.140
15	1.341	1.753	2.131	2.602	2.947	4.073
16	1.337	1.746	2.120	2.583	2.921	4.015
17	1.333	1.740	2.110	2.567	2.898	3.965
18	1.330	1.734	2.101	2.552	2.878	3.922
19	1.328	1.729	2.093	2.539	2.861	3.883
20	1.325	1.725	2.086	2.528	2.845	3.850
21	1.323	1.721	2.080	2.518	2.831	3.819
22	1.321	1.717	2.074	2.508	2.819	3.792
23	1.319	1.714	2.069	2.500	2.807	3.767
24	1.318	1.711	2.064	2.492	2.797	3.745
25	1.316	1.708	2.060	2.485	2.787	3.725
26	1.315	1.706	2.056	2.479	2.779	3.707
27	1.314	1.703	2.052	2.473	2.771	3.690
28	1.313	1.701	2.048	2.467	2.763	3.674
29	1.311	1.699	2.043	2.462	2.756	3.659
30	1.310	1.697	2.042	2.457	2.750	3.646
40	1.303	1.684	2.021	2.423	2.704	3.551
60	1.296	1.671	2.000	2.390	2.660	3.460
120	1.289	1.658	1.980	2.158	2.617	3.373
∞	1.282	1.645	1.960	2.326	2.576	3.291

if the absolute value of the t-test statistics ($|t|$) is greater than the critical value, then the difference is significant.

Paired Sample t-test

$$t = \frac{m}{s/\sqrt{n}}$$

d = the differences between all pairs

m = mean of the difference (d)

s = standard deviation of the difference (d)

n = size of d

$$df = n - 1$$

If the absolute value of the t-test statistics (|t|) is greater than the critical value, then the difference is significant.

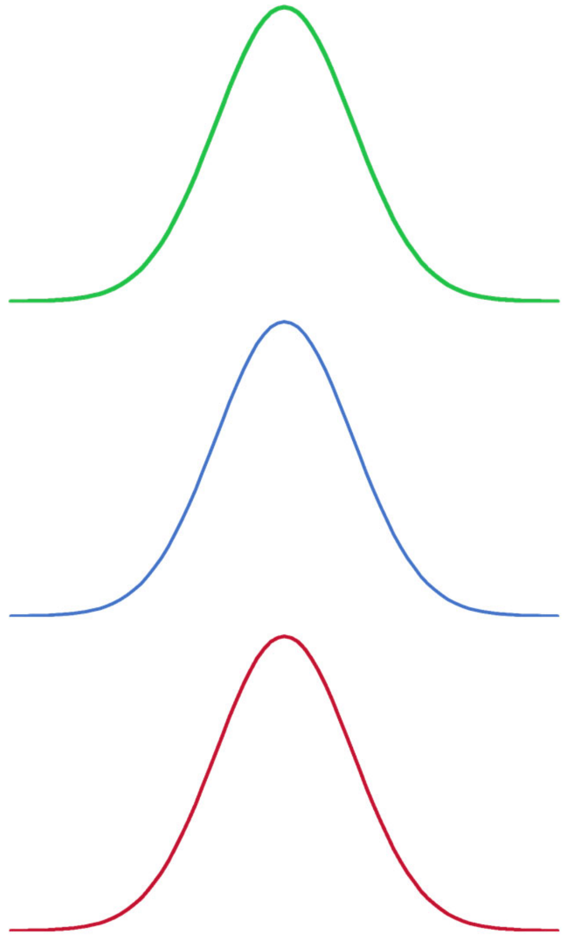
The average of the difference d is compared to 0. If there is any significant difference between the two pairs of samples, then the mean of d is expected to be far from 0.

William Sealy Gosset

- Guinness Brewery
- Differences between barley yields

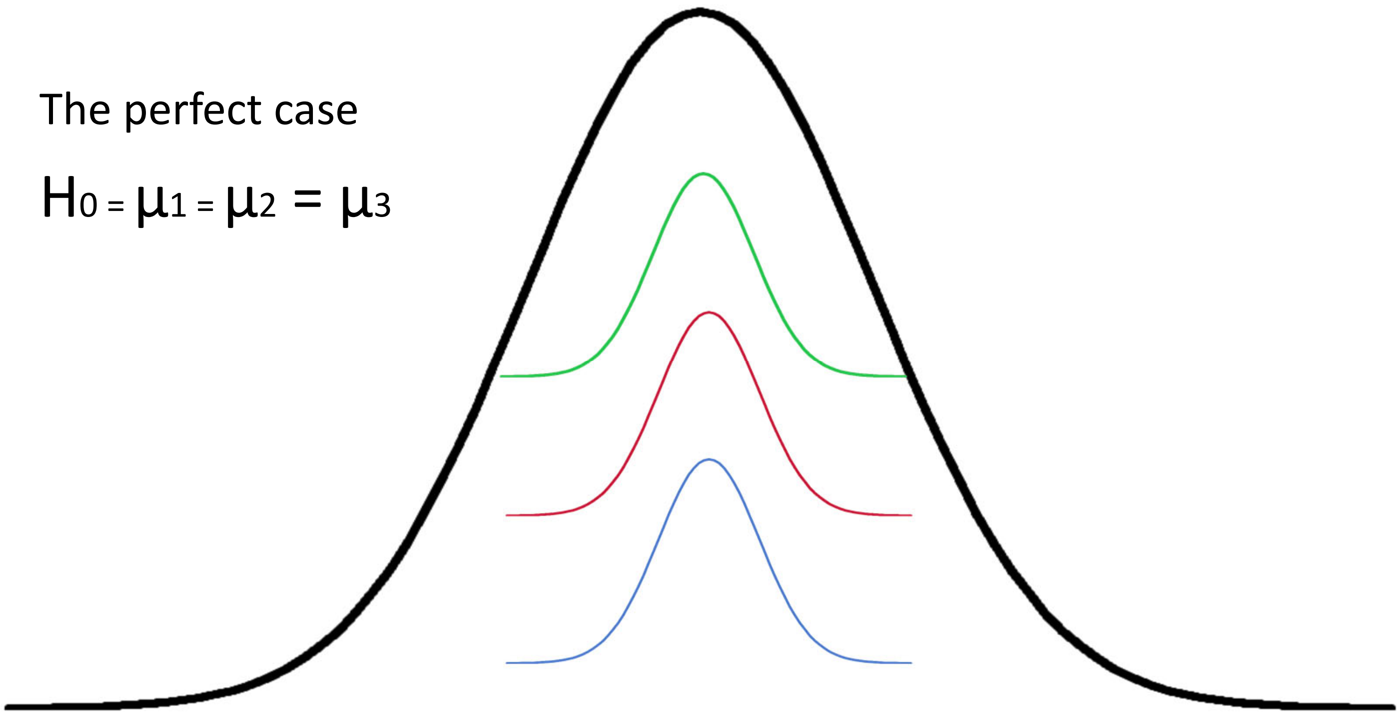
ANOVA

- An Analysis of Variance
  - Compare the means of more than two groups, samples, populations



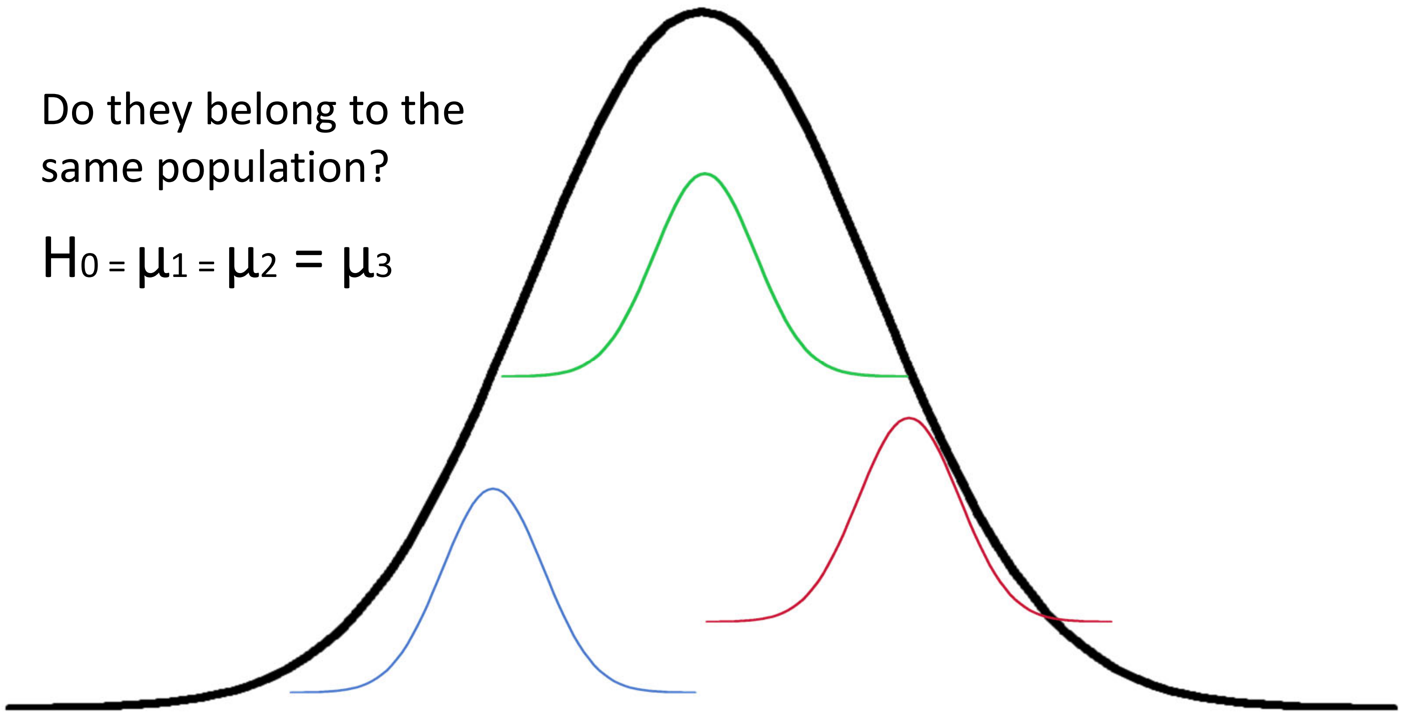
The perfect case

$H_0 = \mu_1 = \mu_2 = \mu_3$



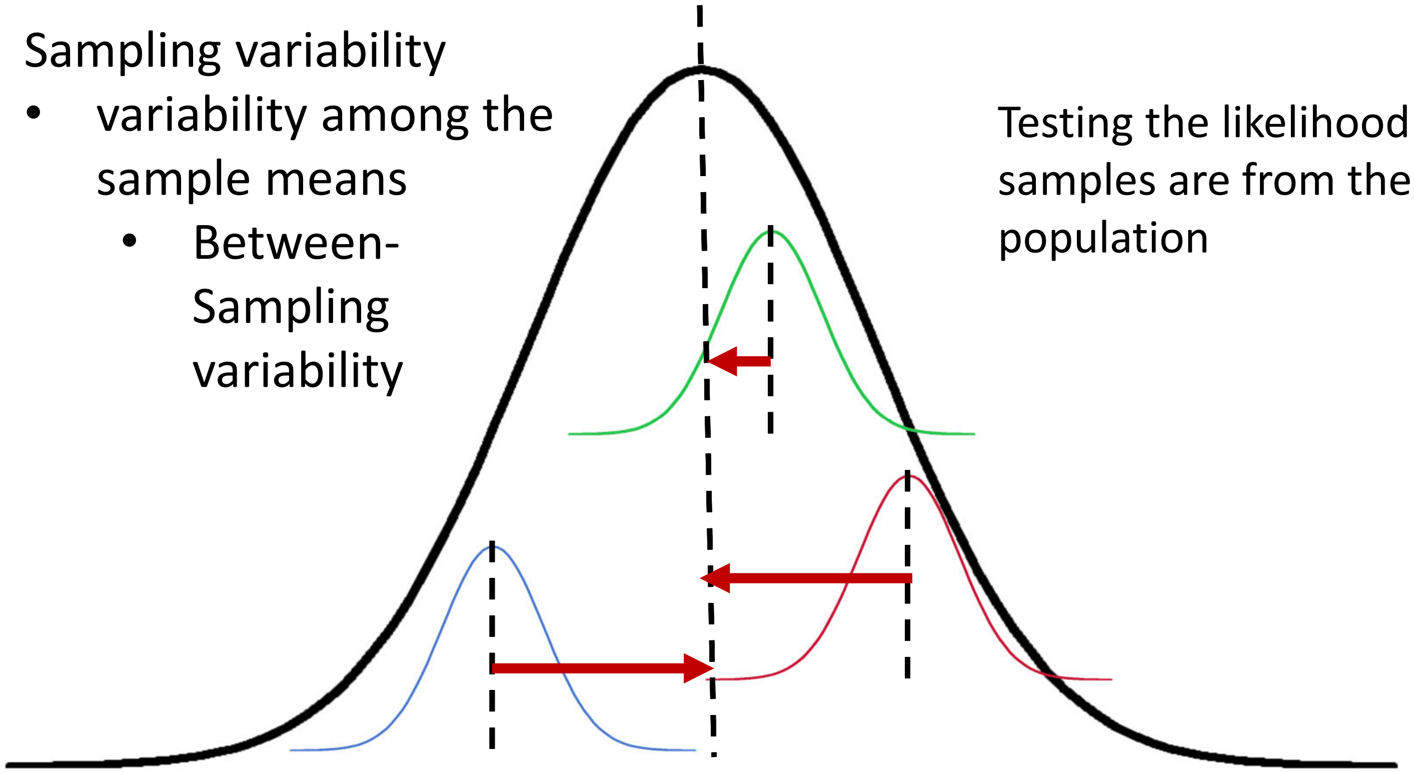
Do they belong to the same population?

$H_0 = \mu_1 = \mu_2 = \mu_3$



Sampling variability

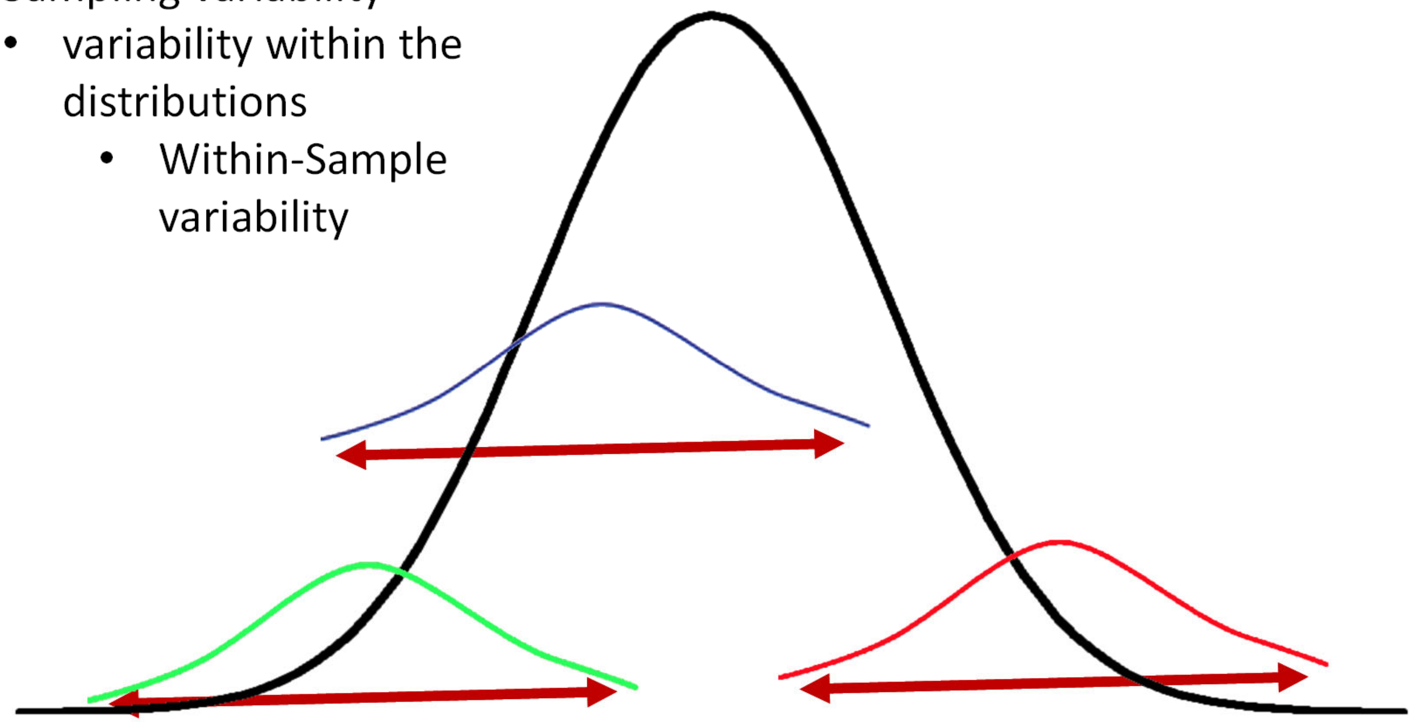
- variability among the sample means
  - Between-Sampling variability



Testing the likelihood that these samples are from the same population

Sampling variability

- variability within the distributions
  - Within-Sample variability

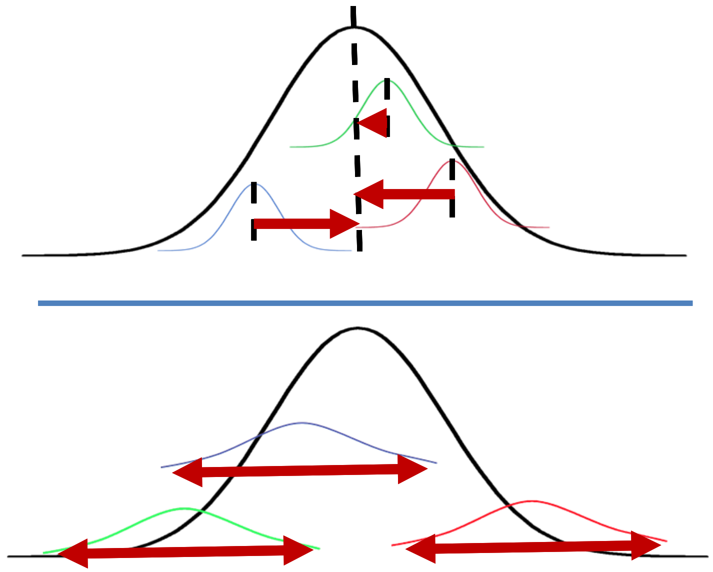


ANOVA =

variability
among the
sample means

---

variability
within the
distributions



Signal

Noise

Independent Sample T-Test: Comparing Two Groups

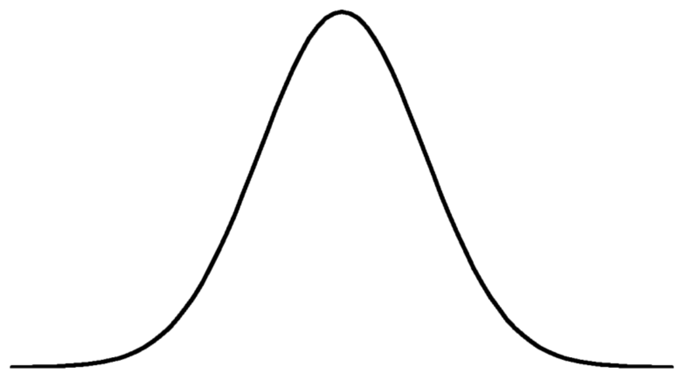
- Professor Huang is teaching two data science classes. One of these classes is delivered online and the other one is delivered in person on campus.

Student	Score
1	88
1	91
1	86
1	92
1	89
1	94
1	87
1	91
1	93
1	88
1	84
1	86
1	86
1	89
1	88
1	90
1	82
1	92
1	85
1	87
2	92
2	95
2	87
2	94
2	86
2	99
2	90
2	93
2	99
2	90
2	85
2	85
2	88
2	96
2	88
2	94
2	88
2	95
2	89
2	86

Steps

- Determine 1-tail or 2-tail test
- Determine if the groups are paired or unpaired
- Determine equal Variance or unequal variance

G1	**G2**
88	92
91	95
86	87
92	94
89	86
94	99
87	90
91	93
93	99
88	90
84	85
86	85
86	88
89	96
88	88
90	94
82	88
92	95
85	89
87	86



G1	G2
88	92
91	95
86	87
92	94
89	86
94	99
87	90
91	93
93	99
88	90
84	85
86	85
86	88
89	96
88	88
90	94
82	88
92	95
85	89
87	86

Determine equal Variance or unequal variance

G1	**G2**
88	92
91	95
86	87
92	94
89	86
94	99
87	90
91	93
93	99
88	90
84	85
86	85
86	88
89	96
88	88
90	94
82	88
92	95
85	89
87	86

```
# read Excel or CSV File
import pandas as pd
# Store Columns as Arrays
import numpy as np

# Perform Independent-Samples T-Test
from scipy.stats import ttest_ind

# Load sample data file
df = pd.read_csv ("Huang Class Differences 2 samples.csv")

# Convert df to Numpy Array
ScoreArray=np.array(df.Score)

# Reshape Numpy Array
ScoreArr = ScoreArray.reshape(2,20)

# Determine Equal Variance by testing if (the Larger Stand Deviation / the smaller Standard
Deviation) > 2
# Assume no equal vaqriance if (the Larger Stand Deviation / the smaller Standard Deviation)
> 2
if ScoreArr[0].std() > ScoreArr[1].std():
    if (ScoreArr[0].std() / ScoreArr[1]) > 2:
        EqualVar = False
    else:
        EqualVar = True
else:
    if (ScoreArr[1].std() / ScoreArr[0].std()) > 2:
        EqualVar = False
    else:
        EqualVar = True

# Obtain T-Stat and Pvalue
SampleT = ttest_ind(ScoreArr[0], ScoreArr[1], equal_var=EqualVar)

SampleT.pvalue
```

Paired Sample T-Test

- Professor Huang wants to know student’s performance before and after taking the course.

Student	Pre-Test	Post-Test
1	51	82
2	47	98
3	36	84
4	54	88
5	51	85
6	51	99
7	66	86
8	63	95
9	68	89
10	36	97
11	46	81
12	64	93
13	41	91
14	64	96
15	50	89
16	46	92
17	62	88
18	64	98
19	35	82
20	57	98

Steps

- Determine 1-tail or 2-tail test
- Determine if the groups are paired or unpaired

Student	Pre-Test	Post-Test
1	51	82
2	47	98
3	36	84
4	54	88
5	51	85
6	51	99
7	66	86
8	63	95
9	68	89
10	36	97
11	46	81
12	64	93
13	41	91
14	64	96
15	50	89
16	46	92
17	62	88
18	64	98
19	35	82
20	57	98

```python
# read Excel or CSV File
import pandas as pd

# Store Columns as Arrays
import numpy as np

# Perform Paired-Samples T-Test
from scipy.stats import ttest_rel

# Load sample data file

df = pd.read_csv ("Huang Class Differences.csv")

# Convert df columns to individual arrays

Student= np.array(df["Student"])
PreTest= np.array(df["Pre-Test"])
PostTest= np.array(df["Post-Test"])

print("student array = ", Student)
print("pre-test array = ", PreTest)
print("post-test array = ", PostTest)

# Compare Means - Paired Samples T-Test

PairedT = ttest_rel(PreTest,PostTest)

print(PairedT)

if PairedT.pvalue < 0.05:
   print("Performamnce of InPerson Students and Online Students are Different")
else:
   print("Performamnce of InPerson Students and Online Students are the same")
```